Abertay University

Thirsty Plant Pi - Mini Project

Tia C

CMP 408 – Systems Internals and Cybersecurity

BSc (Hons) Ethical Hacking

Contents

Introduction

Purpose

The ThirstyPlantPi project's main purpose is to monitor the moisture within the soil each hour, notifying the user if the plant needs watered through Twitter with a picture and status of the plant. The project consists of a Raspberry Pi Zero with a camera and a soil-moisture sensor, an S3 bucket, AWS Lambda service and the Twitter Developer API. The AWS services allow for photos to be stored online and in an accessible format for the lambda service, which saves memory and processing time on the Pi itself.

The ThirstyPlantPi is a scaled down version of IoT within agriculture which is one of the largest applications of IoT in the world. As the population of the world is increasing each year, there are more mouths to feed – "it is essential to guarantee food security", which IoT enabled farming and agriculture can provide (Haque et al., 2021).

The ThirstyPlantPi has been built with security considered throughout the development of the project, particularly as there are credentials being used. In terms of large IoT projects, security is incredibly important as if it fails and the project is compromised, it could potentially cost the farmer thousands or even millions of pounds in crop losses.

Objectives

For the ThirstyPlantPi to work effectively, the following objectives had to be met:

- Software
    - LKM to handle interrupt from tactile button
        - 'Offline' python script to check moisture and turn on LED when LKM triggers python
    - 'IoT' python script to check moisture and send photo to S3 Bucket when soil is dry
        - Cron job to run 'IoT' script each hour
- Hardware
    - Receive input from moisture sensor
        - If plant soil is dry, take photo of thirsty plant
    - Receive input from tactile button
        - Send output to corresponding LED (red if thirsty, green if moist)
- Cloud
    - S3 Bucket to store plant images sent from Pi
    - AWS Lambda triggered by S3 Bucket
        - Lambda function posts most recent image in bucket to Twitter

Procedure

A detailed flow chart demonstrating how the ThirstyPlantPi works can be found in Appendix A. The procedure will be split into two sections, the IoT aspect and the offline aspect of the project.

IoT Procedure

For the IoT aspect of the project, a python script is run each hour via cron job. This script checks the moisture of the soil - if the sensor cannot detect moisture, the camera will take a photo of the plant. This photo is then uploaded to a S3 Bucket, where the AWS Lambda function is triggered. The Lambda function retrieves the most recent object in the bucket, which is the image that has been uploaded and posts to the 'ThirstyPlantPi' Twitter account through the Twitter API.

Python Script

The python script checks the moisture of the soil and depending on the value of the sensor will run either condition of the 'If' statement. If the sensor detects that the soil is wet, it turns the green LED on for two seconds, before performing a clean-up of the GPIO ports to prevent damage. The '*sys.exit()*' function is then run, to stop the execution of the script in a clean way.

If the soil sensor cannot detect moisture, it will begin the notification phase of the script. To start, the external camera attached to the pi, is turned on and takes a photo of the plant – the date and time is added to the top of the photo using the '*camera.annotate_text*' function. The photo is saved locally to the Pi, with the filename being the date and time that the python script was started.

Once the photo is saved locally, the image is uploaded to the S3 bucket on AWS. This is done using the boto3 python library. The boto3 library requires the AWS access key, secret key and session token to access the S3 Bucket, these are stored within the Pi in a credentials file, *'.AWS/credentials'*. This means that the keys are not hardcoded within the python script, which would be a security threat.

When the image is successfully uploaded to the S3 bucket, the image is deleted from the Pi to save memory. The red LED is lit for two seconds, before the GPIO ports are cleaned up and the '*sys.exit()*' function is then run, stopping the script.

S3 Bucket

The S3 bucket is a private bucket that can only be accessed by the python script and the Lambda function. The raspberry pi is set up as an access point within the bucket, so the python script can upload images to the bucket as objects. In the configuration, there is an event notification set up. This event notification can be seen in figure one, with the destination type set to Lambda function where Lambda will carry on the notification process.

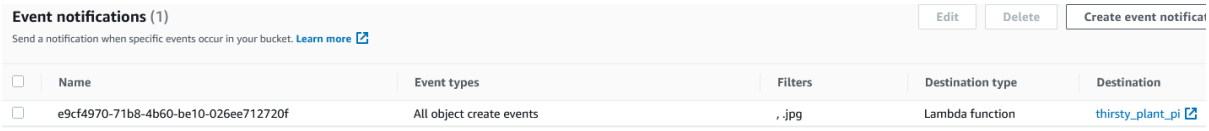


Figure 1 - Event notification within thirstyplantpi S3 Bucket

AWS Lambda

AWS Lambda is responsible for the posting the tweet with the most recent image/object from the S3 Bucket. As seen in figure one previously and again in figure two, the relationship between S3 and Lambda, is that the lambda function is triggered each time a new object is uploaded to the 'thirstyplantpi' bucket.

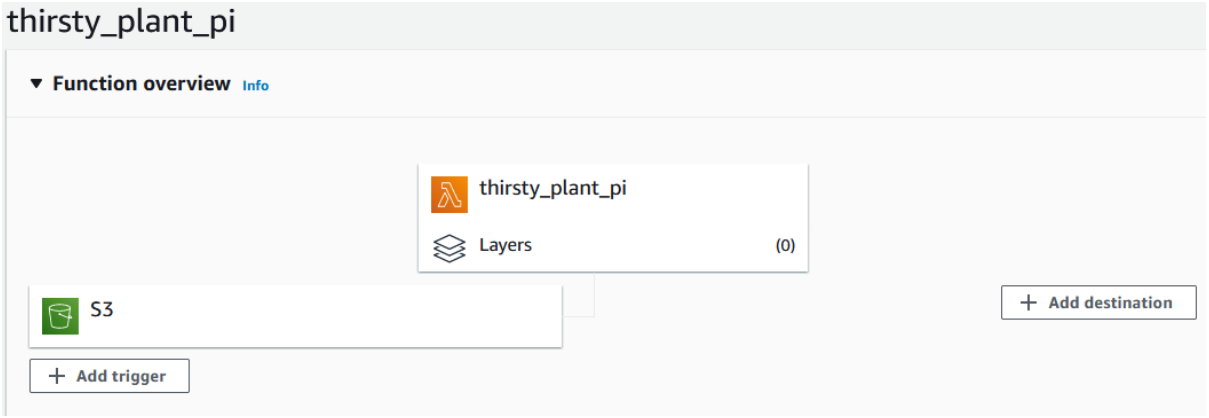


Figure 2 - S3 and Lambda relationship overview

The lambda function is written in python and utilises the boto3 and Tweepy libraries. It also uses the Twitter Developer API. As the function script was over 10MB, it was stored in an S3 bucket as a ZIP file, meaning there are two S3 buckets used within the project. The credentials for the Twitter API and the thirstyplantpi bucket are stored as environment variables within the Lambda configuration, this is seen in figure 3.

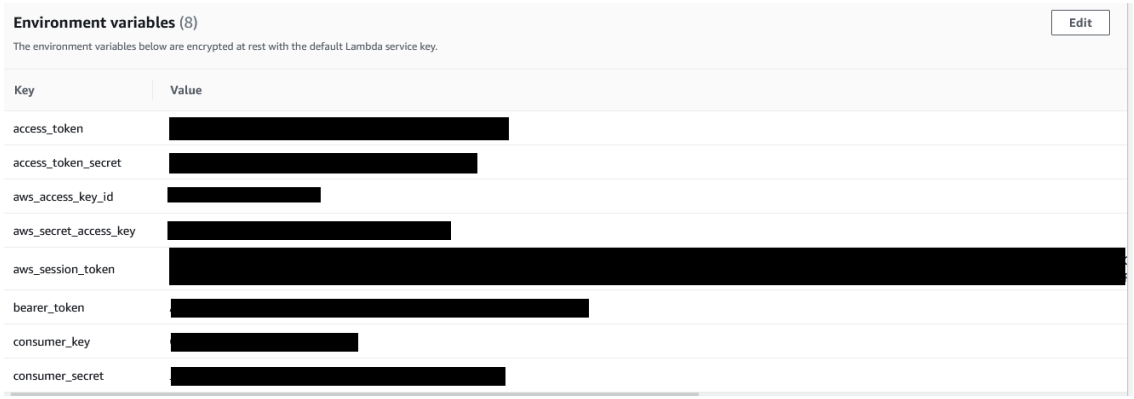


Figure 3 - Environment variables stored within Lambda function configuration

The 'sendTweet()' function of the lambda function sets up the client variable that Tweepy uses to post on behalf of the developer. This function requires the path of where the image has been stored in the lambda function to be passed in as a parameter. The function also requires the Twitter API credentials that are stored within the environment variables. Once the function has access to the Twitter API, a tweet is posted with the most recent image of the plant.

For the ‘sendTweet()’ function to be run, the lambda_handler function is run first. This function will get the most recent object from the bucket that triggered the lambda and download it to a ‘tmp’ folder within the lambda function. Boto3, which was also used in the python script earlier, is used to download the image from the bucket to the lambda function. When the image has been downloaded the ‘sendTweet()’ function is then run, resulting in a tweet notifying the user that the plant is needing watered as can be seen in figure four.

Figure 4 - Screenshot of tweet posted by lambda function with image of plant

Offline Procedure

LKM

For the offline and manual mode of the project, an LKM was created to allow for interrupts by a tactile switch on the breadboard. When the tactile switch is clicked, the LKM will process the interrupt and run the python script to check the moisture level of the soil. As this was done by the user physically pressing the button, there was no need for the IoT/online function of taking a photo and posting a tweet.

Unfortunately, the LKM did not work due to issues with the Makefile – but the code has been included with comments explaining what each line should do. The main aspect of the LKM was that it made use of the ‘*call_usermodehelper()*’ function, that would allow the kernel to execute a user space program without compromising security and functionality.

Python Script

The python script that would be run by the ‘*call_usermodehelper()*’ is a much simpler version of the IoT script. The sensor would check for the moisture in the soil, if there is moisture present – the green LED will be lit for five seconds and then the cleanup and exit

function is executed. If there is no moisture present in the soil, then the red LED will be lit for five seconds and the GPIO cleanup and exit function is run to shut the program down safely.

Conclusion

In conclusion, the project works as planned, apart from the LKM not working – which does affect the offline aspect slightly, as the manual check can still be run through a terminal command. The cloud, hardware and software aspects have been implemented and work effectively together. The cloud services used cost very little, with the project spending $0.40 of the $100 allowance. Security has been considered and implemented throughout the project, particularly in the use of credentials and keys. The project successfully imitates a small-scale IoT solution that could be used within IoT Agriculture for maintaining crops.

Future Work

The project had originally been planned with an automatic irrigation system to be built too, whenever the plant was needing watered, a pump would be activated and water the plant. However, due to time constraints with getting equipment ordered through the university it was not feasible to be completed within the given timeframe. If the author had more time, this would have been implemented, further imitating a small scale IoT agriculture solution.

References

Aruchamy, V., 2022. *How To Write A File Or Data To An S3 Object Using Boto3 - Stack Vidhya*. [online] Stack Vidhya. Available at: <https://www.stackvidhya.com/write-a-file-to-s3-using-boto3/> [Accessed 3 January 2022].

Boto3.amazonaws.com. 2022. *Credentials — Boto3 Docs 1.20.37 documentation*. [online] Available at: <https://boto3.amazonaws.com/v1/documentation/api/latest/guide/credentials.html> [Accessed 2 January 2022].

DEV Community. 2022. *A comprehensive guide for using the Twitter API v2 with Tweepy in Python*. [online] Available at: <https://dev.to/twitterdev/a-comprehensive-guide-for-using-the-twitter-api-v2-using-tweepy-in-python-15d9> [Accessed 3 January 2022].

DEV Community. 2022. *Building a Twitter Bot with Python and AWS Lambda*. [online] Available at: <https://dev.to/jeannienguyen/building-a-twitter-bot-with-python-and-aws-lambda-27jg> [Accessed 3 January 2022].

Elinux.org. 2022. *Raspberry Pi Kernel Compilation - eLinux.org*. [online] Available at: <https://elinux.org/Raspberry_Pi_Kernel_Compilation> [Accessed 13 January 2022].

Giorgio-gross.de. 2022. *Loadable Kernel Modules – The Green Coding Blog*. [online] Available at: <https://www.giorgio-gross.de/blog/index.php/2017/03/18/loadable-kernel-modules/> [Accessed 13 January 2022].

Gist. 2022. *a simple linux driver code to register GPIO as button input to trigger LED*. [online] Available at: <https://gist.github.com/itrobotics/224a0c549ae073ce991c> [Accessed 13 January 2022].

GitHub. 2022. *GitHub - mistylackie/raspberry-pi-camera-to-aws-s3: Code that takes pictures with the Raspberry Pi camera module and uploads those photos to an S3 bucket*. [online] Available at: <https://github.com/mistylackie/raspberry-pi-camera-to-aws-s3> [Accessed 3 January 2022].

Haque, M., Haque, S., Sonal, D., Kumar, K. and Shakeb, E., 2021. Security Enhancement for IoT Enabled Agriculture. *Materials Today: Proceedings*,.

module, E. and Matveychikov, I., 2022. *Execute shell command in kernel module*. [online] Stack Overflow. Available at: <https://stackoverflow.com/questions/11193648/execute-shell-command-in-kernel-module> [Accessed 13 January 2022].

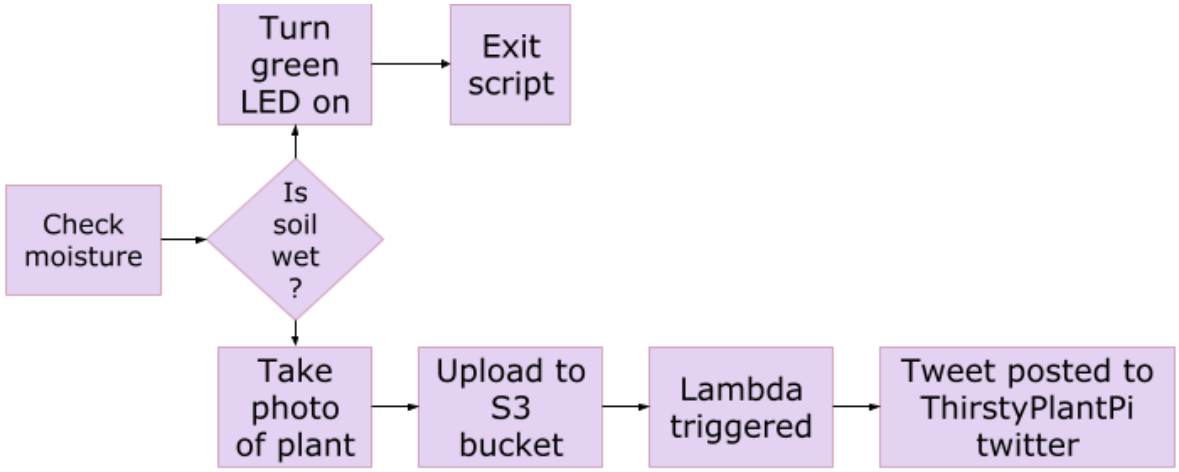
Molloy, D., 2022. *Writing a Linux Loadable Kernel Module (LKM) - Interfacing to GPIOs | derekmolloy.ie*. [online] derekmolloy.ie. Available at: <http://derekmolloy.ie/kernel-gpio-programming-buttons-and-leds/#Example_1_Button_Press_LED_Light_LKM> [Accessed 13 January 2022].

Technology News Wales. 2022. [online] Available at: <https://businessnewswales.com/how-iot-technology-is-helping-to-revolutionise-the-agriculture-sector/> [Accessed 14 January 2022].

Appendix

Appendix A

Online Methodology



Offline Methodology